```
MMM         MMM  PPPPPPPPPPPP
MMM         MMM  PPPPPPPPPPPP
MMM         MMM  PPPPPPPPPPPP
MMMMMM   MMMMMM  PPP         PPP
MMMMMM   MMMMMM  PPP         PPP
MMMMMM   MMMMMM  PPP         PPP
MMM   MMM   MMM  PPP         PPP
MMM   MMM   MMM  PPP         PPP
MMM   MMM   MMM  PPP         PPP
MMM         MMM  PPPPPPPPPPPP
MMM         MMM  PPPPPPPPPPPP
MMM         MMM  PPPPPPPPPPPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
MMM         MMM  PPP
```

```
                                        E  4
**FILE**ID**MPINT






MM      MM  PPPPPPPP      IIIIII    NN      NN  TTTTTTTTTT
MM      MM  PPPPPPPP      IIIIII    NN      NN  TTTTTTTTTT
MMMM  MMMM  PP      PP      II      NN      NN      TT
MMMM  MMMM  PP      PP      II      NN      NN      TT
MM  MM  MM  PP      PP      II      NNNN    NN      TT
MM  MM  MM  PP      PP      II      NNNN    NN      TT
MM      MM  PPPPPPPP        II      NN  NN  NN      TT
MM      MM  PPPPPPPP        II      NN  NN  NN      TT
MM      MM  PP              II      NN    NNNN      TT
MM      MM  PP              II      NN    NNNN      TT
MM      MM  PP              II      NN      NN      TT          ....
MM      MM  PP              II      NN      NN      TT          ....
MM      MM  PP            IIIIII    NN      NN      TT          ....
MM      MM  PP            IIIIII    NN      NN      TT          ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000   ;
0000   ; Version:      'V04-000'
0000   ;
0000
0000           .MCALL  MFPR
0000           .TITLE  MPINT - MULTI-PROCESSOR INTERRUPT HANDLER
0000           .IDENT  'V04-000'
0000
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;
0000
0000   ;++
0000   ;
0000   ; Facility:  Executive , Hardware fault handling
0000   ;
0000   ; Abstract:  This module contains the VAX multiport memory interrupt handler.
0000   ;
0000   ; Environment: MODE=Kernel,  Interrupt
0000   ;
0000   ; Author:  RICHARD I. HUSTVEDT, Creation date:  15-May-1979
0000   ;
0000   ; Modified by:
0000   ;
0000   ;       V03-007 KDM0026         Kathleen D. Morse       14-Oct-1982
0000   ;               Conditionalize time-out logic based on debugging switch
0000   ;               so that taking a breakpoint on the secondary does not
0000   ;               make the primary turn it off.
0000   ;
0000   ;       V03-006 KDM0018         Kathleen D. Morse       13-Oct-1982
0000   ;               Add logic to primary code for secondary wait check
0000   ;               request.
0000   ;
0000   ;       V03-005 KDM0020         Kathleen D. Morse       04-Oct-1982
0000   ;               Add time-out logic to primary code that requests the
0000   ;               secondary to do an invalidate of a system space address.
```

```
MPI
Sym

BU
BU
BU
EM
EM
EM
EM
EM
EM
EM
EM
EM
ER
ER
EX
EX
EX
IP
LC
MA
MM
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
MP
```

```
                    0000     53 ;
                    0000     54 ;         V03-004 KDM0012        Kathleen D. Morse       20-Sep-1982
                    0000     55 ;                 Add second error log buffer and clear MA780 interrupt
                    0000     56 ;                 before checking for reason of interrupt.
                    0000     57 ;
                    0000     58 ; 01      -
                    0000     59 ;--
```

```
                0000          .SBTTL  DEFINITIONS
                0000    ;
                0000    ; INCLUDE FILES:
                0000    ;
                0000
                0000    ;
                0000    ; MACROS:
                0000    ;
                0000
                0000    ;
                0000    ; EQUATED SYMBOLS:
                0000    ;
                0000          $BUGDEF                         ; Bugcheck indicator bit definitions
                0000          $EMBDEF                         ; Error log message buffer definitions
                0000          $EMBDEF SS                      ; Sys srv err log message buffer defs
                0000          $IPLDEF                         ; Interrupt priority levels
                0000          $LCKDEF                         ; Interlock bit definitons
                0000          $MPMDEF                         ; Multiport memory registers
                0000          $MPSDEF                         ; Secondary processor state definitions
                0000          $PCBDEF                         ; Define PCB offsets
                0000          $PHDDEF                         ; Define process header offsets
                0000          $PRDEF                          ; Define processor registers
                0000          $PTEDEF                         ; Define page table entry format
                0000          $RPBDEF                         ; Define reboot parameter block
                0000          $VADEF                          ; Virtual address definition
                0000
     00000004   0000    MAX_PORTS = 4                         ; Maximum number of ports
                0000
                0000    ;
                0000    ; OWN STORAGE:
                0000    ;
            00000000          .PSECT  AEXENONPAGED,LONG       ;
```

```
                                0000    94          .SBTTL  MPS$MAINIT - INITIALIZE MULTI-PORT MEMORY ADAPTER
                                0000    95 ;++
                                0000    96 ;
                                0000    97 ; FUNCTIONAL DESCRIPTION:
                                0000    98 ;
                                0000    99 ; This routine is called at system initialization and after a power
                                0000   100 ; recovery restart to initialize the port adapter by clearing any
                                0000   101 ; errors and enabling all interrupts.
                                0000   102 ;
                                0000   103 ;
                                0000   104 ; OUPUTS:
                                0000   105 ;
                                0000   106 ;        Any errors in port are cleared and all interrupts are enabled.
                                0000   107 ;
                                0000   108 ;--
                                0000   109 MPS$MAINIT::                                      ;
                     3F   BB    0000   110          PUSHR   #^M<R0,R1,R2,R3,R4,R5>    ; Save registers
         54   0000'CF     D0    0002   111          MOVL    W^MPS$AL_MPMBASE,R4       ; Get base of MPM registers
           00400000 8F    D0    0007   112          MOVL    #MPM$M_CSR_PU,-           ; Clear any power-up status
                     64         000D   113                  MPM$L_CSR(R4)             ;
                          D0    000E   114          MOVL    #MPM$M_CR_ERRS!-          ; Clear any port errors and
                                000F   115                  MPM$M_CR_MIE,-            ; Enable master interrupt
  04 A4    FF000001 8F          000F   116                  MPM$L_CR(R4)              ;
                          D0    0016   117          MOVL    #MPM$M_SR_SS!-            ; Clear any status errors and
                                0017   118                  MPM$M_SR_IDL!-            ;  disable error interrupts
                                0017   119                  MPM$M_SR_IT!-             ;
                                0017   120                  MPM$M_SR_AGP!-            ;
                                0017   121                  MPM$M_SR_MXF!-            ;
                                0017   122                  MPM$M_SR_ACA,-            ;
  08 A4    D000E000 8F          0017   123                  MPM$L_SR(R4)              ;
             50    0C A4  D0    001E   124          MOVL    MPM$L_INV(R4),R0          ; Get invalidation register
     50    800FFFFF 8F    CA    0022   125          BICL    #^C<MPM$M_INV_STADR>,R0   ; Clear all but starting address
                                0029   126          ASSUME  MPM$V_INV_ID EQ 0         ; Cached nexus id's start at 0
    0C A4      50   01    C9    0029   127          BISL3   #1@0,R0,MPM$L_INV(R4)     ; Set cpu (nexus 0) as cached
                          D0    002E   128          MOVL    #MPM$M_ERR_ELR!-          ; Clear any errors
                                002F   129                  MPM$M_ERR_IMP,-           ;
  10 A4    90000000 8F          002F   130                  MPM$L_ERR(R4)             ;
           00000400 8F    D0    0036   131          MOVL    #MPM$M_CSR1_MIA,-         ; Clear any error
                  18 A4         003C   132                  MPM$L_CSR1(R4)            ;
                  1C A4   D4    003E   133          CLRL    MPM$L_MR(R4)              ; Clear any diagnostic settings
              50     64   D0    0041   134          MOVL    MPM$L_CSR(R4),R0          ; Get CSR register
                     00   EF    0044   135          EXTZV   #MPM$V_CSR_PORT,-         ; Get port number
       50     50     02         0046   136                  #MPM$S_CSR_PORT,R0,R0     ;
              50     04   C4    0049   137          MULL    #MAX_PORTS,R0             ; Compute interrupt enable bit #
              50     10   C0    004C   138          ADDL    #MPM$V_IIE_CTL,R0         ; ...
    24 A4     0F     50   78    004F   139          ASHL    R0,#^XF,MPM$L_IIE(R4)     ; Enable interport interrupts
                                0054   140                                            ;  from all ports
                     3F   BA    0054   141          POPR    #^M<R0,R1,R2,R3,R4,R5>    ; Restore registers
                          05    0056   142          RSB                               ; Return
                                0057   143
```

```
                                  0057          .SBTTL  MPS$INTPRIM - INTERRUPT PRIMARY PROCESSOR
                                  0057  ;++
                                  0057  ; FUNTIONAL DESCRIPTION:
                                  0057  ;
                                  0057  ; MPS$INTPRIM is called to cause an interrupt to the primary processor.
                                  0057  ;
                                  0057  ; CALLING SEQUENCE:
                                  0057  ;
                                  0057  ;       BSB/JSB MPS$INTPRIM
                                  0057  ;
                                  0057  ; INPUT PARAMETERS:
                                  0057  ;
                                  0057  ;       NONE
                                  0057  ;
                                  0057  ; OUTPUT PARAMETERS:
                                  0057  ;
                                  0057  ;       NONE
                                  0057  ;--
                                  0057
                                  0057  MPS$INTPRIM::                           ;
        0000'DF    0000'CF    D0  0057          MOVL    W^MPS$GL_PRIMSKT,@W^MPS$GL_MPMIIR ; Trigger primary interrupt
                              05  005E          RSB                             ; And return
                                  005F
                              005F          .SBTTL  MPS$INTSCND - INTERRUPT SECONDARY PROCESSOR
                              005F     ;++
                              005F     ; FUNCTIONAL DESCRIPTION:
                              005F     ;
                              005F     ; MPS$INTSCND is called to interrupt the secondary processor.
                              005F     ;
                              005F     ; CALLING SEQUENCE:
                              005F     ;
                              005F     ;        BSB/JSB MPS$INTSCND
                              005F     ;
                              005F     ; INPUT PARAMETERS:
                              005F     ;
                              005F     ;        NONE
                              005F     ;
                              005F     ; OUTPUT PARAMETERS:
                              005F     ;
                              005F     ;        NONE
                              005F     ;--
                              005F
                              005F     MPS$INTSCND::
 0000'DF    0000'CF       D0  005F          MOVL    W^MPS$GL_SCNDMSKT,@W^MPS$GL_MPMIIR ; Trigger secondary interrupt
                          05  0066          RSB                                        ; And return
                              0067
```

```
                           0067          .SBTTL  MPS$PINTSR - PRIMARY PROCESSOR INTERRUPT SERVICE ROUTINE
                           0067  ;++
                           0067  ; FUNCTIONAL DESCRIPTION:
                           0067  ;
                           0067  ; MPS$PINTSR is entered via the interrupt vector for the MA780 in
                           0067  ; the primary processor in response to a call to MPS$INTPRIM.
                           0067  ;
                           0067  ;--
                           0067
                           0067          .ALIGN  LONG
                           0068  MPS$PINTSR::                            ; Primary interrupt service routine
                  50    DD 0068          PUSHL   R0                      ; Save R0
       50   0000'CF     D0 006A          MOVL    W^MPS$AL_MPMBASE,R0     ; Get base of MPM registers
    20 A0   0000'CF     D0 006F          MOVL    W^MPS$GL_PRIMSKC,MPM$L_IIR(R0)  ; Clear pending interrupt
                  50  8ED0 0075          POPL    R0                      ; Restore R0
    04 0000'CF    00    E7 0078          BBCCI   #MPS$V_SECBUGCHK,W^MPS$GL_SECREQFLG,10$ ; Br if no bugchk to do
                           007E          .LIST   MEB
                           007E          BUG_CHECK MPBADMCK,FATAL        ; Jump to bugcheck code
                      FEFF 007E                  .WORD   ^XFEFF
                     0004' 0080                  .IIF IDN <FATAL>,<FATAL> , .WORD        BUG$_MPBADMCK!4
                  00000080 0082  MPS$GW_BUGCHKCOD == .-2                 ; Location for secondary to place the
                           0082                                          ;  type of bugcheck it is requesting
                           0082          .NLIST  MEB
    46 0000'CF    01    E7 0082  10$:    BBCCI   #MPS$V_SECERRLOG,W^MPS$GL_SECREQFLG,50$ ; Br if no errlog to do
                  3F    BB 0088          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers
    2F 0000'CF    00    E7 008A          BBCCI   #MPS$V_ERLBUF1,W^MPS$GL_ERLBUFIND,30$ ; Br if no entry in buf 1
       53   0000'CF     9E 0090          MOVAB   W^MPS$AL_ERLBUF1,R3     ; Get address of error log entry
         51    FC A3    3C 0095  20$:    MOVZWL  EMB$W_SIZE(R3),R1       ; Find size of error log entry
       55    51   10    C3 0099          SUBL3   #EMB$C_HD_LENGTH,R1,R5  ; Remember size of entry to move in
        00000000'GF     16 009D          JSB     G^ERL$ALLOCEMB          ; Allocate an error log buffer
               19 50    E9 00A3          BLBC    R0,30$                  ; Br if none available
               62 63    D0 00A6          MOVL    EMB$L_HD_SID(R3),EMB$L_HD_SID(R2) ; Set system ID in error msg
     04 A2   04 A3      B0 00A9          MOVW    EMB$W_HD_ENTRY(R3),EMB$W_HD_ENTRY(R2) ; Set msg type in errlog
                  52    DD 00AE          PUSHL   R2                      ; Remember address of buffer
  10 A2    10 A3  55    28 00B0          MOVC3   R5,EMB$C_HD_LENGTH(R3),EMB$C_HD_LENGTH(R2) ; Move msg into buf
                  52  8ED0 00B6          POPL    R2                      ; Restore buffer address
        00000000'GF     16 00B9          JSB     G^ERL$RELEASEMB         ; Release the error log buffer
    07 0000'CF    01    E7 00BF  30$:    BBCCI   #MPS$V_ERLBUF2,W^MPS$GL_ERLBUFIND,40$ ; Br if no entry in buf 2
       53   0000'CF     9E 00C5          MOVAB   W^MPS$AL_ERLBUF2,R3     ; Get address of error log entry
                  C9    11 00CA          BRB     20$                     ; Join common code
                  3F    BA 00CC  40$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
                           00CE  ;
                           00CE  ; Nothing to be done at device IPL.  This is either a spurious
                           00CE  ; interrupt, or an event flag wait check request from the secondary,
                           00CE  ; or a legitimate reschedule request from the secondary.  Cause the
                           00CE  ; reschedule software interrupt and check for requested work at that
                           00CE  ; IPL.
                           00CE  ;
                           00CE  50$:    SOFTINT #5                      ; Request IPL 5 interrupt
                        02 00D1          REI                             ; And return
                           00D2
```

```
                              00D2          .SBTTL  MPS$SINTSR - SECONDARY INTERRUPT SERVICE ROUTINE
                              00D2      ;++
                              00D2      ; FUNCTIONAL DESCRIPTION:
                              00D2      ;
                              00D2      ; MPS$SINTSR is entered in response to an interrupt on the secondary
                              00D2      ; processor.  The interrupt was sent for one of the following reasons:
                              00D2      ;
                              00D2      ;       1) An AST was sent to the process currently
                              00D2      ;          running on the secondary
                              00D2      ;          (Primary processor is executing QAST.)
                              00D2      ;
                              00D2      ;       2) A system space address was invalidated by
                              00D2      ;          the primary processor
                              00D2      ;          (Primary processor is executing FREWSL or PAGEFAULT.)
                              00D2      ;
                              00D2      ;       3) The primary wants to bugcheck.
                              00D2      ;
                              00D2      ; The secondary processor, not knowing which reason the interrupt
                              00D2      ; was sent, does the appropriate work to handle all the reasons.
                              00D2      ; (Since the code is small, there is no need to figure out the real
                              00D2      ; reason for the interrupt.)  The following list corresponds to the
                              00D2      ; work done to handle the above conditions causing an interrupt:
                              00D2      ;
                              00D2      ;       1) The ASTLVL for the process currently running
                              00D2      ;          on the secondary is updated
                              00D2      ;
                              00D2      ;       2) An invalidate is done for the system space
                              00D2      ;          address indicated by MPS$GL_INVALID
                              00D2      ;
                              00D2      ;       3) First, fold up the current process.
                              00D2      ;          Second, load the loop address into the RPB.
                              00D2      ;          Third, acknowlege the bugcheck request.
                              00D2      ;          Fourth, halt to turn off mapping.  Execution continues
                              00D2      ;          if restart is enabled, by the console program executing
                              00D2      ;          RESTAR.CMD.
                              00D2      ;--
                              00D2
                              00D2          .ALIGN  LONG
                              00D4      MPS$SINTSR::                            ; Secondary interrupt service routine
                     50    DD 00D4          PUSHL   R0                          ; Save R0
       50   0000'CF        D0 00D6          MOVL    W^MPS$AL_MPMBASE,R0         ; Get base of MPM registers
    20 A0   0000'CF        D0 00DB          MOVL    W^MPS$GL_SCNDMSKC,MPM$L_IIR(R0); Clear pending interrupt
                     50  8ED0 00E1          POPL    R0                          ; Restore R0
  00 0000'CF       00      E6 00E4          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,5$ ; Flush cache queue
  26 0000'CF       00      E0 00EA  5$:     BBS     #BUG$V_BUGCHK,W^MPS$GL_BUGCHECK,10$ ; Br if bugcheck requested
  4B 0000'CF       00      E0 00F0          BBS     #MPS$V_STOPREQ,W^MPS$GL_STOPFLAG,50$ ; Br if STOP/CPU requested
                              00F6      ;
                              00F6      ; Update the ASTLVL for the process currently running on the secondary.
                              00F6      ;
                     50    DD 00F6          PUSHL   R0                          ; Save R0
       50   0000'CF        D0 00F8          MOVL    W^MPS$GL_CURPCB,R0          ; Get current PCB address
          50     6C A0     D0 00FD          MOVL    PCB$L_PHD(R0),R0            ; Get PHD address
       50   00CF C0        90 0101          MOVB    PHD$B_ASTLVL(R0),R0         ; And fetch ASTLVL
            13     50      DA 0106          MTPR    R0,#PR$_ASTLVL              ; Update current value
                              0109      ;
                              0109      ; Invalidate the system space address that is contained in MPS$GL_INVALID.
                              0109      ;
```

```
                               0109         INVALID W^MPS$GL_INVALID         ; Invalidate requested page
                  0000'CF   D4 010E         CLRL    W^MPS$GL_INVALID         ; And acknowledge it
                       50 BEDO 0112         POPL    R0                       ; Restore R0
                            02 0115         REI                              ; And continue
                               0116
                               0116    ;
                               0116    ; Primary processor has requested a bugcheck.  The secondary must fold
                               0116    ; up the process it is running and loop quietly in a safe place out of
                               0116    ; the way of the primary.
                               0116    ;
         00 0000'CF   00    E6 0116    10$:     BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,20$ ; Flush cache queue
            04   0000'CF    D1 011C    20$:     CMPL    W^MPS$GL_STATE,#MPS$K_EXECSTATE ; Was LDPCTX done?
                      01    12 0121             BNEQ    30$                      ; Br if not done, don't do SVPCTX
                      07       0123             SVPCTX                           ; Save state of current process
         50   00000000'GF   D0 0124    30$:     MOVL    G^EXE$GL_RPB,R0          ; Get address of RPB
00FC C0  60   00000100 8F   C1 012B             ADDL3   #RPB$B_WAIT,RPB$L_BASE(R0),RPB$L_BUGCHK(R0) ; Load loop adr
            0000'CF   06    D0 0135             MOVL    #MPS$K_STOPSTATE,W^MPS$GL_STATE ; Indicate processor not active
         00 0000'CF   01    E6 013A             BBSSI   #BUG$V_ACK1,W^MPS$GL_BUGCHECK,40$ ; Acknowlege bugcheck request
                            00 0140    40$:     HALT                             ; This halt causes the secondary to
                               0141                                              ; start executing RESTAR.CMD on the
                               0141                                              ; console device if restart is enabled.
                               0141    ;
                               0141    ; A STOP/CPU was issued.  The secondary must return its current process,
                               0141    ; if any, load a wait loop into the RPB, and halt.
                               0141    ;
         00 0000'CF   00    E6 0141    50$:     BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,60$ ; Flush cache queue
            04   0000'CF    D1 0147    60$:     CMPL    W^MPS$GL_STATE,#MPS$K_EXECSTATE ; Is there a current process?
                      06    12 014C             BNEQ    70$                      ; Br if no current state to save
                      07       014E             SVPCTX                           ; Save state of current process
            0000'CF   02    D0 014F             MOVL    #MPS$K_DROPSTATE,W^MPS$GL_STATE ; Primary must take process back
         50   00000000'GF   D0 0154    70$:     MOVL    G^EXE$GL_RPB,R0          ; Get address of RPB
00FC C0  60   00000100 8F   C1 015B             ADDL3   #RPB$B_WAIT,RPB$L_BASE(R0),RPB$L_BUGCHK(R0) ; Load loop adr
                  0000'CF   D4 0165             CLRL    W^MPS$GL_INVALID         ; Indicate no invalidate to wait on
         00 0000'CF   01    E6 0169             BBSSI   #MPS$V_STOPACK1,W^MPS$GL_STOPFLAG,80$ ; Acknowlege STOP request
                            00 016F    80$:     HALT                             ; Stop the secondary
```

```
                              0170              .SBTTL  MPS$INVALID - Relay invalidate request to secondary
                              0170      ;++
                              0170      ; FUNCTIONAL DESCRIPTION:
                              0170      ;
                              0170      ; MPS$INVALID relays a translation buffer invalidate request to
                              0170      ; the secondary processor and waits for acknowledgement before
                              0170      ; proceeding.  Since P0 pages are only referenced by the processor
                              0170      ; currently executing a process, only system pages need to be
                              0170      ; invalidated by both the primary and secondary processors at
                              0170      ; the same time.
                              0170      ;
                              0170      ; This code is hooked into the pagefault exception handling code.
                              0170      ;
                              0170      ;--
                              0170
                              0170      MPS$INVALID::                                 ;
    03 A3    84 8F        8A  0170              BICB    #<PTE$M_VALID!PTE$M_MODIFY>@-24,3(R3); Clear valid and modify
                              0175                                                    ; (Replaced instruction)
                              0175              INVALID R2                            ; Invalidate for primary processor
       38 52    1F        E1  0178              BBC     #VA$V_SYSTEM,R2,60$           ; Only invalidate for system space
                              017C              ASSUME  MPS$K_STOPSTATE GT MPS$K_INITSTATE
 00 0000'CF     00        E6  017C              BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,10$ ; Flush cache queue
    05   0000'CF          D1  0182      10$:    CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Secondary active?
                    2B    18  0187              BGEQ    60$                           ; Br if no, secondary not responding
    0000'CF         52    D0  0189              MOVL    R2,W^MPS$GL_INVALID           ; Set address to invalidate
                              018E
                              018E              .IF     DF,MPPFMSWT
                              018E              INCL    W^PFM$L_CNT_INVAL             ; Add one to perf meas invalidate ctr
                              018E              .ENDC
                              018E
               FECE       30  018E              BSBW    MPS$INTSCND                   ; Interrupt secondary processor
                 5A       DD  0191              PUSHL   R10                           ; Save R10
5A    00E4E1C0 8F         D0  0193              MOVL    #15000000,R10                 ; Initialize time-out counter
            02 5A         F4  019A      20$:    SOBGEQ  R10,30$                       ; Repeat loop, waiting for secondary ack
                              019D
                              019D              .IF     NDF,MPDBGSWT
                 1B       11  019D              BRB     70$                           ; Go log failure and turn off secondary
                              019F              .IFF    ;MPDBGSWT DEFINED
                              019F              BRB     30$                           ; Don't turn off secondary, just loop
                              019F                                                    ;  if debugging as breakpoints would
                              019F                                                    ;  cause the secondary to get turned off
                              019F              .ENDC
                              019F
 00 0000'CF     00        E6  019F      30$:    BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,40$ ; Flush cache queue
          0000'CF         D5  01A5      40$:    TSTL    W^MPS$GL_PFAILTIM             ; Has secondary powerfailed?
                 06       12  01A9              BNEQ    50$                           ; Br if yes, don't wait for him
                              01AB
                              01AB              .IF     DF,MPPFMSWT
                              01AB              INCL    W^PFM$L_CNT_IWAIT             ; Inc perf meas invalidate loop counter
                              01AB              .ENDC
                              01AB
          0000'CF         D5  01AB              TSTL    W^MPS$GL_INVALID              ; Acknowledged yet?
                 E9       12  01AF              BNEQ    20$                           ; No, continue waiting
                 5A     8ED0  01B1      50$:    POPL    R10                           ; Restore R10
      00000000'GF         17  01B4      60$:    JMP     G^MMG$FRE_TRYSKIP             ; Continue with page fault
                              01BA
                              01BA      ;
```

```
                                01BA  ; The secondary did not acknowledge the invalidate request.  Therefore,
                                01BA  ; the primary assumes it has died.  A message is placed in the error log
                                01BA  ; and an indicator is incremented showing that this failure occurred.
                                01BA  ; Then the multi-processing code is unhooked from the running system,
                                01BA  ; making the primary ignore any further activity from the secondary.
                                01BA  ; The pool space containing the multi-processing code is left untouched
                                01BA  ; just in case the secondary is eventually resurrected and tries to
                                01BA  ; continue executing.  If this happens, some unexpected interrupt will
                                01BA  ; probably be logged by the primary but nothing will have been lost,
                                01BA  ; except whatever process the secondary may have been running.
                                01BA  ;
                                01BA  ; This design allows a gradual degradation of the system to a single
                                01BA  ; processor 11/780, instead of forcing a bugcheck.
                                01BA  ;
                  0000'CF   D6  01BA  70$:    INCL    W^MPS$GL_INV_NACK           ; Indicate secondary did not acknowledge
                       3F   BB  01BE          PUSHR   #^M<R0,RT,R2,R3,R4,R5> ; Save registers for MOVC
              51    00'8F   9A  01C0          MOVZBL  #MPS$C_INV_NACK,R1          ; Size of ASCII message text
                   51   15  C0  01C4          ADDL    #<EMB$K_SS_LENGTH+3>,R1 ; Add in overhead for message
                   51   03  CA  01C7          BICL    #3,R1                       ; Buffer size modulo 4
              00000000'GF   16  01CA          JSB     G^ERL$ALLOCEMB              ; Allocate error log buffer
                   1E   50  E9  01D0          BLBC    R0,80$                      ; If failure, just unhook MP code
                04 A2   27  B0  01D3          MOVW    #EMB$C_SS,EMB$W_SS_ENTRY(R2) ; Set type of error log message
         10 A2    0000'8F   B0  01D7          MOVW    #MPS$C_INV_NACK,EMB$W_SS_MSGSZ(R2) ; Set size of ASCII text msg
                        52  DD  01DD          PUSHL   R2                          ; Save buffer address
 12 A2    0000'CF  0000'8F  28  01DF          MOVC    #MPS$C_INV_NACK,W^MPS$T_INV_NACK,EMB$B_SS_MSGTXT(R2) ; Msg txt
                      52  8ED0  01E8          POPL    R2                          ; Restore buffer address
              00000000'GF   16  01EB          JSB     G^ERL$RELEASEMB             ; Release error log buffer
                                01F1
                                01F1  ;
                                01F1  ; Now unhook the multi-processing code and restore the system to
                                01F1  ; a single processor 11/780, vanilla VMS system.
                                01F1  ;
      00 00000000'EF    00  E6  01F1  80$:    BBSSI   #MPS$V_STOPREQ,MPS$GL_STOPFLAG,90$ ; Indic primary forced a stop
           5A  00000000'GF  D0  01F9  90$:    MOVL    G^EXE$GL_MP,R10             ; Get address of MP code
                     FDFD'  30  0200          BSBW    W^MPS$UNHOOK                ; Unhook MP code from VMS code
                  043F 8F   BA  0203          POPR    #^M<R0,R1,R2,R3,R4,R5,R10> ; Restore registers
                     FFAA   31  0207          BRW     60$                         ; Continue with normal VMS code
```

```
                               020A             .SBTTL  MPS$BUGCHECK - Relay bugcheck request to secondary and wait
                               020A      ;++
                               020A      ; FUNCTIONAL DESCRIPTION:
                               020A      ;
                               020A      ; MPS$BUGCHECK makes sure that the secondary is out of the way before
                               020A      ; the primary procedes with the bugcheck logic.  It sets a flag to
                               020A      ; indicate a bugcheck is requested.  Then interrupts the secondary to
                               020A      ; make it notice the flag.  The primary then waits for the secondary
                               020A      ; to acknowlege the bugcheck request.
                               020A      ;
                               020A      ; ENVIRONMENT:
                               020A      ;
                               020A      ;        Executed by the primary processor.
                               020A      ;        IPL = 31
                               020A      ;
                               020A      ;--
                               020A
                               020A      MPS$BUGCHECK::                                  ;
 00 0000'CF    00       E6     020A             BBSSI   #BUG$V_BUGCHK,W^MPS$GL_BUGCHECK,10$ ; Indicate bugcheck request
                               0210             ASSUME  MPS$K_STOPSTATE GT MPS$K_INITSTATE
    05    0000'CF       D1     0210      10$:   CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Is secondary active?
                 1B     18     0215             BGEQ    50$                             ; Br on not active, don't request bugchk
 00 0000'CF    00       E6     0217             BBSSI   #BUG$V_BUGCHK,W^MPS$GL_BUGCHECK,20$ ; Indicate bugcheck request
               FE3F     30     021D      20$:   BSBW    W^MPS$INTSCND                   ; Interrupt secondary to notice request
50    00E4E1C0 8F       D0     0220             MOVL    #15000000,R0                    ; Wait a significant amount of time
 02 0000'CF    01       E7     0227      30$:   BBCCI   #BUG$V_ACK1,W^MPS$GL_BUGCHECK,40$ ; Wait for secondary acknowlege
                 03     11     022D             BRB     50$                             ; Secondary done, continue with bugchk
              F5 50     F4     022F      40$:   SOBGEQ  R0,30$                          ; Repeat as secondary not acknowleged
        00000000'9F     17     0232      50$:   JMP     @#EXE$INIBOOTADP                ; Continue with normal bugcheck code
```

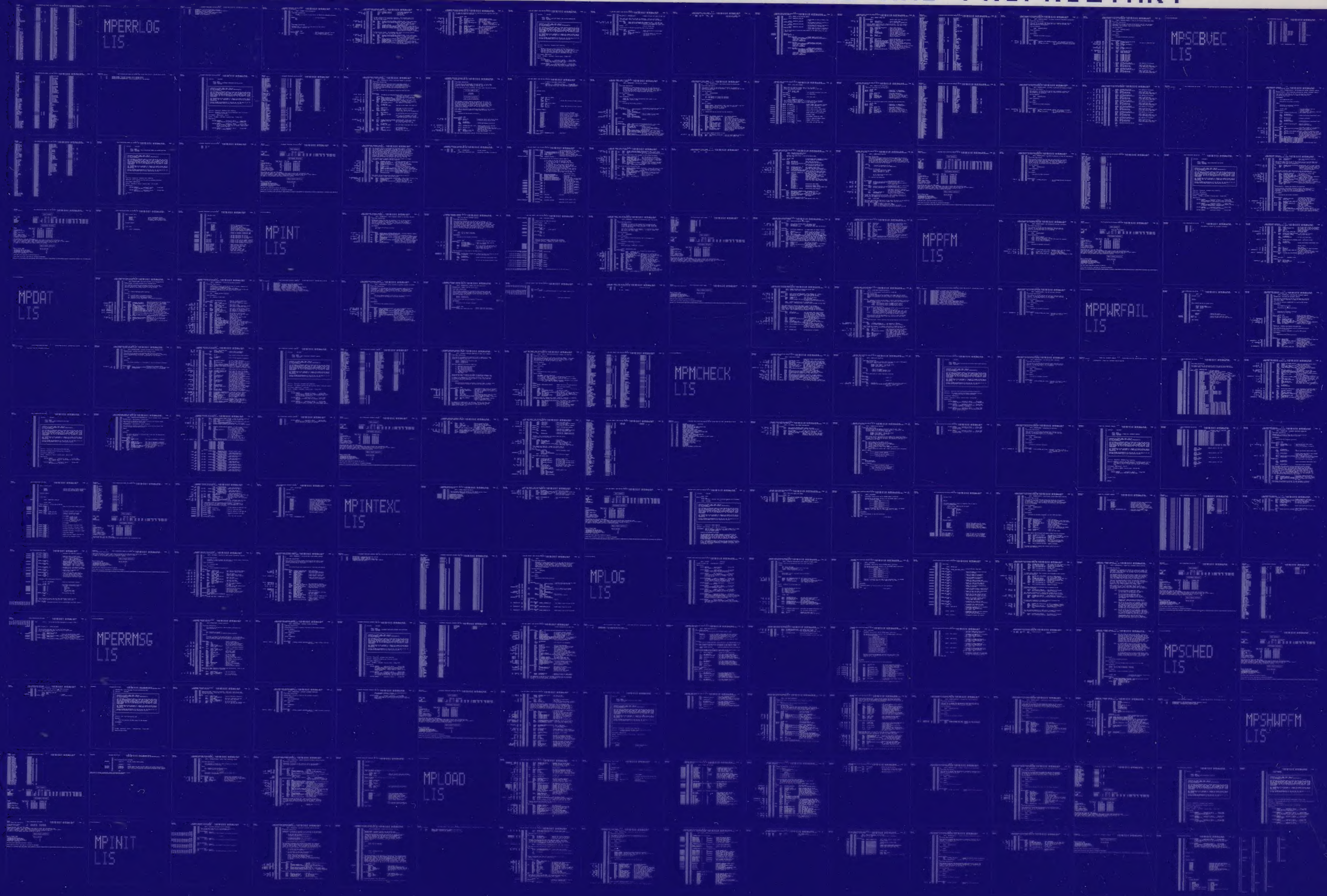
```
                                    0238          .SBTTL  MPS$SECBUGCHK - Relay secondary's bugcheck request to primary
                                    0238  ;++
                                    0238  ; FUNCTIONAL DESCRIPTION:
                                    0238  ;
                                    0238  ; MPS$SECBUGCHK is executed when the secondary processor wants to initiate
                                    0238  ; a bugcheck.   It sets a flag indicating a bugcheck is requested and
                                    0238  ; interrupts the primary to make it notice the flag.  The secondary then
                                    0238  ; waits for the primary to interrupt it with the actual bugcheck request
                                    0238  ; by executing a self-branch.
                                    0238  ;
                                    0238  ; INPUTS:
                                    0238  ;
                                    0238  ;       The return address pushed on the stack by calling this routine
                                    0238  ;       is the address of the bugcheck code being requested.
                                    0238  ;
                                    0238  ; OUTPUTS:
                                    0238  ;
                                    0238  ;       None
                                    0238  ;
                                    0238  ; ENVIRONMENT:
                                    0238  ;
                                    0238  ;       Executed by the secondary processor.
                                    0238  ;
                                    0238  ;--
                                    0238
                                    0238  MPS$SECBUGCHK::                                       ;
          FE42 CF     00 BE    B0   0238          MOVW    @(SP),W^MPS$GW_BUGCHKCOD ; Set type of bugcheck requested
          50    00000000'GF    D0   023E          MOVL    G^EXE$GL_RPB,R0         ; Get address of RPB
 00FC C0  60    00000100 8F    C1   0245          ADDL3   #RPB$B_WAIT,RPB$L_BASE(R0),RPB$L_BUGCHK(R0) ; Load loop adr
                                    024F          SETIPL  #IPL$_SYNCH              ; Lower IPL, enabling inter-proc intrpt
          00 0000'CF      00   E6   0252          BBSSI   #MPS$V_SECBUGCHK,W^MPS$GL_SECREQFLG,10$ ; Set request flag
                        FDFC   30   0258  10$:    BSBW    W^MPS$INTPRIM            ; Interrupt primary processor
          00 0000'CF      00   E6   025B          BBSSI   #LCK$V_INTERLOCK,W^MPS$GL_INTERLOCK,20$ ; Flush cache queue
             05   0000'CF      D1   0261  20$:    CMPL    W^MPS$GL_STATE,#MPS$K_INITSTATE ; Secondary active?
                          02   18   0266          BGEQ    40$                      ; Br if not active
                          FE   11   0268  30$:    BRB     30$                      ; Wait for interrupt from primary to
                                    026A                                           ;  handle the bugcheck
                               00   026A  40$:    HALT                             ; This halt causes the secondary to
                                    026B                                           ; start executing RESTAR.CMD on the
                                    026B                                           ; console device if restart is enabled.
                                    026B
                                    026B          .END
```

```
BUG$V_ACK1          = 00000001            MPS$GL_INV_NACK       ********   X   02
BUG$V_BUGCHK        = 00000000            MPS$GL_MPMIIR         ********   X   02
BUG$_MPBADMCK         ********   X   02   MPS$GL_PFAILTIM       ********   X   02
EMB$B_SS_MSGTXT     = 00000012            MPS$GL_PRIMSKC        ********   X   02
EMB$C_HD_LENGTH     = 00000010            MPS$GL_PRIMSKT        ********   X   02
EMB$C_SS            = 00000027            MPS$GL_SCNDMSKC       ********   X   02
EMB$K_SS_LENGTH     = 00000012            MPS$GL_SCNDMSKT       ********   X   02
EMB$L_HD_SID        = 00000000            MPS$GL_SECREQFLG      ********   X   02
EMB$W_HD_ENTRY      = 00000004            MPS$GL_STATE          ********   X   02
EMB$W_SIZE          = FFFFFFFC            MPS$GL_STOPFLAG       ********   X   02
EMB$W_SS_ENTRY      = 00000004            MPS$GW_BUGCHKCOD    = 00000080 RG    02
EMB$W_SS_MSGSZ      = 00000010            MPS$INTPRIM           00000057 RG    02
ERL$ALLOCEMB          ********   X   02   MPS$INTSCND           0000005F RG    02
ERL$RELEASEMB         ********   X   02   MPS$INVALID           00000170 RG    02
EXE$GL_MP             ********   X   02   MPS$K_DROPSTATE     = 00000002
EXE$GL_RPB            ********   X   02   MPS$K_EXECSTATE     = 00000004
EXE$INIBOOTADP        ********   X   02   MPS$K_INITSTATE     = 00000005
IPL$_SYNCH          = 00000008            MPS$K_STOPSTATE     = 00000006
LCK$V_INTERLOCK     = 00000000            MPS$MAINIT            00000000 RG    02
MAX_PORTS           = 00000004            MPS$PINTSR            00000068 RG    02
MMG$FRE_TRYSKIP       ********   X   02   MPS$SECBUGCHK         00000238 RG    02
MPM$L_CR            = 00000004            MPS$SINTSR            000000D4 RG    02
MPM$L_CSR           = 00000000            MPS$T_INV_NACK        ********   X   02
MPM$L_CSR1          = 00000018            MPS$UNHOOK            ********   X   02
MPM$L_ERR           = 00000010            MPS$V_ERLBUF1       = 00000000
MPM$L_IIE           = 00000024            MPS$V_ERLBUF2       = 00000001
MPM$L_IIR           = 00000020            MPS$V_SECBUGCHK     = 00000000
MPM$L_INV           = 0000000C            MPS$V_SECERRLOG     = 00000001
MPM$L_MR            = 0000001C            MPS$V_STOPACK1      = 00000001
MPM$L_SR            = 00000008            MPS$V_STOPREQ       = 00000000
MPM$M_CR_ERRS       = FF000000            PCB$L_PHD           = 0000006C
MPM$M_CR_MIE        = 00000001            PHD$B_ASTLVL        = 000000CF
MPM$M_CSR1_MIA      = 00000400            PR$_ASTLVL          = 00000013
MPM$M_CSR_PU        = 00400000            PR$_IPL             = 00000012
MPM$M_ERR_ELR       = 10000000            PR$_SIRR            = 00000014
MPM$M_ERR_IMP       = 80000000            PR$_TBIS            = 0000003A
MPM$M_INV_STADR     = 7FF00000            PTE$M_MODIFY        = 04000000
MPM$M_SR_ACA        = 80000000            PTE$M_VALID         = 80000000
MPM$M_SR_AGP        = 10000000            RPB$B_WAIT          = 00000100
MPM$M_SR_IDL        = 00004000            RPB$L_BASE          = 00000000
MPM$M_SR_IT         = 00008000            RPB$L_BUGCHK        = 000000FC
MPM$M_SR_MXF        = 40000000            VA$V_SYSTEM         = 0000001F
MPM$M_SR_SS         = 00002000
MPM$S_CSR_PORT      = 00000002
MPM$V_CSR_PORT      = 00000000
MPM$V_IIE_CTL       = 00000010
MPM$V_INV_ID        = 00000000
MPS$AL_ERLBUF1        ********   X   02
MPS$AL_ERLBUF2        ********   X   02
MPS$AL_MPMBASE        ********   X   02
MPS$BUGCHECK          0000020A RG    02
MPS$C_INV_NACK        ********   X   02
MPS$GL_BUGCHECK       ********   X   02
MPS$GL_CURPCB         ********   X   02
MPS$GL_ERLBUFIND      ********   X   02
MPS$GL_INTERLOCK      ********   X   02
MPS$GL_INVALID        ********   X   02
```

```
                                                   +----------------+
                                                   ! Psect synopsis !
                                                   +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
AEXENONPAGED                    0000026B  (   619.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                                +------------------------+
                                                ! Performance indicators !
                                                +------------------------+

Phase                        Page faults    CPU Time        Elapsed Time
-----                        -----------    --------        ------------
Initialization                        32    00:00:00.10     00:00:01.15
Command processing                   131    00:00:00.85     00:00:04.28
Pass 1                               288    00:00:08.89     00:00:28.05
Symbol table sort                      0    00:00:01.20     00:00:01.96
Pass 2                               109    00:00:02.14     00:00:06.64
Symbol table output                   12    00:00:00.09     00:00:00.38
Psect synopsis output                  2    00:00:00.02     00:00:00.23
Cross-reference output                 0    00:00:00.00     00:00:00.00
Assembler run totals                 576    00:00:13.29     00:00:42.70

The working set limit was 1500 pages.
47795 bytes (94 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 791 non-local and 32 local symbols.
507 source lines were read in Pass 1, producing 17 object records in Pass 2.
30 pages of virtual memory were used to define 29 macros.

                                               +--------------------------+
                                               ! Macro library statistics !
                                               +--------------------------+

Macro library name                             Macros defined
------------------                             --------------
_$255$DUA28:[MP.OBJ]MP.MLB;1                               4
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                            17
_$255$DUA28:[SYSLIB]STARLET.MLB;2                          5
TOTALS (all libraries)                                    26

950 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:MPINT/OBJ=OBJ$:MPINT MSRC$:MPPREFIX/UPDATE=(ENH$:MPPREFIX)+MSRC$:MPINT/UPDATE=(ENH$:MPINT)+EXECML$/LIB+LIB$:MP.MLB/LI
```

0248 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



MPERRLOG
LIS

MPSCBVEC
LIS

MPINT
LIS

MPPFM
LIS

MPDAT
LIS

MPPWRFAIL
LIS

MPMCHECK
LIS

MPINTEXC
LIS

MPLOG
LIS

MPERRMSG
LIS

MPSCHED
LIS

MPSHWPFM
LIS

MPLOAD
LIS

MPINIT
LIS